[0086] FIG. 10 shows parallel process selection method used in a source program of the CPU1 and the CPU1A.

[0087] A C-language source program undergoes processing by a preprocessor before being compiled. In a source program, the preprocessor performs processing using a preprocessor statement indicated as "#", and an instruction to perform compiling is then given to a compiler.

[0088] FIG. 10 shows a source program written in C language in which it is possible to instruct the compiler to perform compiling using the preprocessor statement (#pragma). In the case of (a), the function "kansu1" and the function "kansu2" are compiled two in parallel and four in parallel respectively, where

#pragma parallel4 (kansu1) and

#pragma parallel2 (kansu2)

[0089] Furthermore, in the case of (b), the descriptions in between #pragma parallel2 and #pragma parallel2end are compiled two in parallel (in this case, a while loop).

[0090] As similar to the case of the CPU1 shown in FIG. 1, in the case where four processing details are described in an instruction code of 128 bits, when the instruction code is compiled two in parallel, fields corresponding to two predetermined arithmetic processing blocks are not manipulated. Using an instruction decode block DU, the no-operation fields cause a stop signal STPi to be asserted on the corresponding arithmetic processing blocks so as to stop the arithmetic processing manipulation.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-092661
(43)Date of publication of application : 06.04.2001

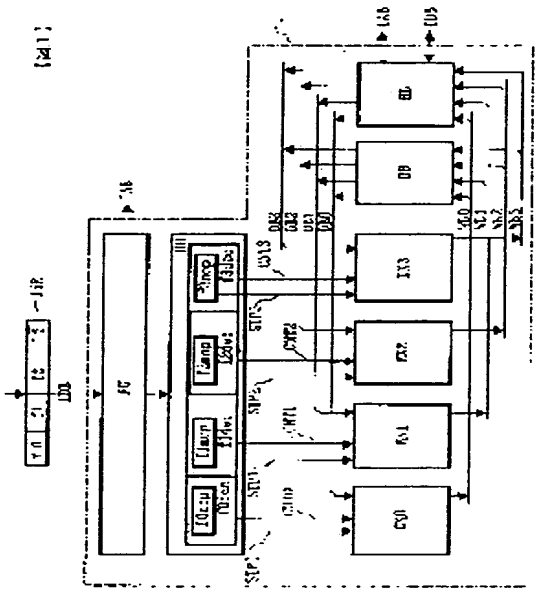
(51)Int.Cl. G06F 9/38
G06F 1/32
G06F 1/04
G06F 9/30
G06F 9/45

(21)Application number : 11-267950
(22)Date of filing : 22.09.1999

(71)Applicant : HITACHI LTD
(72)Inventor : MITSUISHI NAOMIKI

(54) DATA PROCESSOR

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a data processor for controlling the parallelism of parallel processing, and for easily operating power consumption control corresponding to the parallelism without increasing any logical scale.
SOLUTION: A CPU 1 capable of parallel arithmetic processing by using plural executing parts (EX0-EX3) by decoding read instructions stops the operation clock signal of any executing part in a non-operational state, and inhibits any data input or output when the number of arithmetic processing to be executed in parallel is smaller than the number of executing parts at the time of operating the parallel arithmetic processing. In this case, an instruction code itself to designate the contents of the arithmetic processing at the executing parts is provided with information indicating whether or not the processing should be executed in parallel. Thus, it is possible to reduce any undesired power consumption as necessary.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－92661
（P2001－92661A）
(43)公開日　平成13年４月６日(2001.4.6)

(51)Int.Cl.'		識別記号	ＦＩ		テーマコード'(参考)
Ｇ０６Ｆ	9/38	３７０	Ｇ０６Ｆ 9/38	３７０Ａ	５Ｂ０１１
	1/32		1/04	３０１Ｃ	５Ｂ０１３
	1/04	３０１	9/30	３３０Ｂ	５Ｂ０３３
	9/30	３３０		３５０Ｆ	５Ｂ０７９
		３５０	1/00	３３２Ｂ	５Ｂ０８１

審査請求　未請求　請求項の数８　ＯＬ　（全 16 頁）　最終頁に続く

(21)出願番号　特願平11－267950

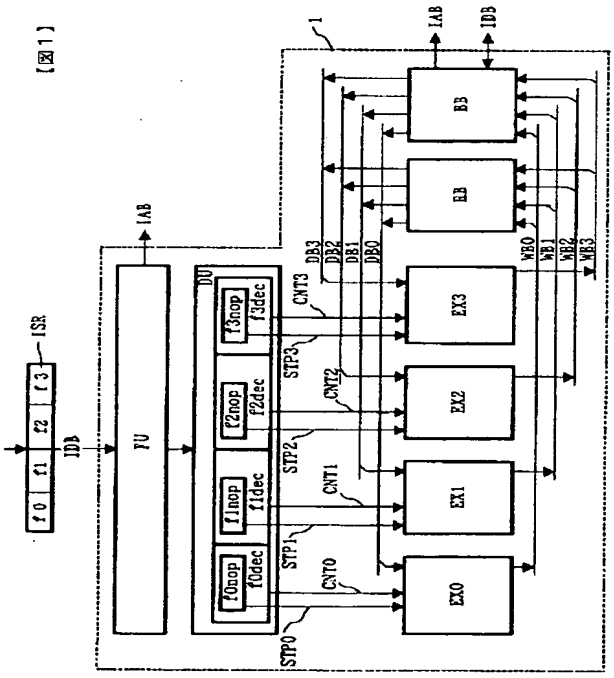
(22)出願日　平成11年９月22日(1999.9.22)

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　三ッ石　直幹
東京都小平市上水本町五丁目20番１号　株式会社日立製作所半導体グループ内
(74)代理人　100089071
弁理士　玉村　静世

最終頁に続く

(54)【発明の名称】　データ処理装置

(57)【要約】
【課題】　並列処理の並列度が制御可能であってその並列度に応じた消費電力制御を、論理的規模を増大させず、容易に行なえるデータ処理装置を提供する。
【解決手段】　読み込んだ命令を解読し複数の実行部（ＥＸ０～ＥＸ３）を用いて並列演算処理可能なＣＰＵ（１）は、並列演算処理を行なう場合に、実行部の数より、並列実行すべき演算処理が少ないとき、動作しない実行部の動作クロック信号を停止すると共にデータ入出力などを禁止する。実行部での演算処理内容を指定する命令コードそれ自体に、並列実行すべき処理であるか否かを示す情報を持たせる。これにより、不所望な電力消費を、随時、抑制することができる。



1

【特許請求の範囲】

【請求項１】　並列処理を行なうデータ処理装置であって、

命令をリードする命令フェッチ部と、並列動作可能であって夫々演算器を内蔵した複数の実行部と、前記命令フェッチ部がリードした命令を解読して得られる制御信号に基づいて前記複数個の実行部の動作を制御する制御部と、を有し、

前記夫々の実行部は、前記制御部の出力する所定の制御信号に応答して個別的に動作停止状態を採り得るものであることを特徴とするデータ処理装置。

【請求項２】　前記制御部は、命令コード中に含まれ、前記実行部の処理を指定する情報のフィールドに対し、前記処理の無操作を判定したとき、対応する実行部の動作を停止状態に制御する前記所定の制御信号を生成するものであることを特徴とする請求項１記載のデータ処理装置。

【請求項３】　前記制御部は、所定の命令コードを解読したとき、これに続く命令の実行を並列させる所定の制御信号を複数の実行部に出力するものであることを特徴とする請求項１記載のデータ処理装置。

【請求項４】　演算動作停止させる一部の実行部を指定する手段を更に有し、前記制御部は、その手段で指定された実行部を、命令の並列実行対象から除外するものであることを特徴とする請求項３記載のデータ処理装置。

【請求項５】　前記指示する手段は前記実行部によってアクセスされるレジスタ手段であることを特徴とする請求項４記載のデータ処理装置。

【請求項６】　前記実行部は、第１のクロック信号に同期動作される演算器を有し、前記制御部からの所定の制御信号に応答して前記第１のクロック信号の変化を抑止して演算動作を停止させるゲート手段を有して成るものであることを特徴とする請求項１乃至５の何れか１項に記載のデータ処理装置。

【請求項７】　異なる分周比のクロック信号を生成する分周器と、分周器の出力を選択するセレクタと、セレクタによる選択の制御情報が設定されるクロック選択レジスタとを有し、

前記第１のクロック信号は前記セレクタで選択されたクロック信号に基いて生成されるものであることを特徴とする請求項６記載のデータ処理装置。

【請求項８】　請求項１乃至７の何れか１項に記載のデータ処理装置が実行するオブジェクトプログラムを生成する開発装置であって、ソースプログラムのステップ関数またはファイルの少なくとも一つの単位で前記実行部の並列度数を指定する手段を有し、これによって指定された並列度数を前記単位でオブジェクトプログラムに反映するものであることを特徴とする開発装置。

【発明の詳細な説明】

【０００１】

2

【発明の属する技術分野】本発明は、マイクロコンピュータ、マイクロコントローラ、データプロセッサ等のデータ処理装置に関し、例えば、単一の半導体チップに形成された、機器制御用（機器組込み用途）のデータ処理装置に利用して有効な技術に関するものである。

【０００２】

【従来の技術】機器制御用のマイクロコンピュータには、所謂リアルタイム性が要求される。即ち、制御対象機器の制約から、所定のイベントに呼応した処理を、所定の時間内に完了し、制御を行なう必要がある。また、システム全体の時間的な制約の中で、所要の処理を完了させる必要がある。更に、イベントは同時に発生する場合もあるから、このような場合、各イベントに呼応した処理に対して、夫々の制約を満足しなければならない。データ処理全体の処理時間ではなく、イベント毎に、個別の処理の時間を短縮する必要があるから、例えば、１個のＣＰＵによるような、単一の命令ストリームに対応する処理性能を向上させる必要がある。

【０００３】例えば、ＯＳ（オペレーティングシステム）を使用する場合には、ＯＳの処理プログラム実行中には、割込みが禁止されることが一般的であるが、これは、イベントに対する処理開始を遅らせることになるから、機器制御やリアルタイム処理においては好ましくない。このため、機器制御用のＯＳにはリアルタイムＯＳなどと呼ばれる、割込み禁止時間を短縮したり、所要の優先順位の割込みは常に受け付けられるようにしたものが用いられる。

【０００４】データ処理装置において、単一の命令ストリームに対応するように、１個のＣＰＵで、いわゆる並列処理を行ない、実行時間を短縮可能にするものとして、平成７年１月（株）日経ＢＰ社発行『日経エレクトロニクス』ｐｐ６８～８０「１９９８年に転機、ハードウェアを単純化してＶＬＩＷへ」などに記載されているように、スーパスカラやＶＬＩＷ（Very Long Instruction Word）などがある。いずれも演算処理ブロックを複数持ち（例えば４個）、同時に実行できる処理の数を増やして（例えば４並列）、全体的な処理性能を向上している。

【０００５】前記ＶＬＩＷは、１２８ビットなどの長い固定長の命令コードを持ち、演算処理ブロックに夫々対応する各フィールドに、処理内容を指定するようになっている。コンパイル時に、ソースプログラムを解析して、予め並列実行できる処理を、１つのＶＬＩＷの命令に構成する。

【０００６】一方、スーパスカラは、実行時に、リードした個々の命令を解析して、並列実行可能な命令を選択して、並列処理を行なう。この場合、コンパイル時に、ソースプログラムを解析して、並列処理を制御する命令を挿入し、この制御命令に基づいて、続く複数命令を並列処理する例もある。

【０００７】
【発明が解決しようとする課題】しかしながら、上記の並列処理では、複数の処理を、常に実行可能であるとは限らない。例えば、データの依存関係によるものがあり、所定のライトの後にリードを行なうべき場合に、並列処理を行なうと、この順序を保てなくなることなどがあるからである。並列処理を行なえない場合には、一部の演算処理ブロックは使用されない。
【０００８】また、データ処理装置の応用分野は広がっているから、携帯型などバッテリで動作するようなものなど、処理速度とともに、低消費電力が必要になるものもあり、実使用上、常に最大の処理速度が必要であるとは限らない。
【０００９】消費電力を低減する為に、ＣＰＵ内部の機能ブロックのうち、当該命令で使用する或いは使用しない機能ブロックへのクロック信号供給を許可／禁止する例が、特開平１０－３４０１２７号公報に記載されている。この例では、使用状態に応じて、処理速度を優先したり、消費電力を優先したりすることには対応できない。また、命令や処理速度などについては言及されていない。
【００１０】一方、特開平９－１８５５８９号公報には、複数のタスクを並列に処理可能な複数のプロセッサを有し、タスクを構成する処理単位毎のプロセッサ資源の要求に応答してタスク管理手段が、休止状態のプロセッサを発生させるように、複数のプロセッサの内の特定のプロセッサにプロセッサ資源の不足が生じない範囲でタスクを割り付け、電源供給制御手段により、休止状態のプロセッサに対する電源供給を停止し、システム全体の消費電力を節減可能にしたデータ処理システムが例示されている。
【００１１】この例では、各タスクが個別のプロセッサに割当てられるから、各タスクの処理時間は、プロセッサ全体の休止状態には依存せず、一定である。複数のプロセッサは、所要の処理を高速化する為に用いられてはいない。
【００１２】また、回路の一部に電源の供給を停止するような場合は、電源が供給されている部分と、供給されていない部分のインタフェース信号に、電気的に不都合が生じないように、例えば、貫通電流が生じたり、トランジスタに不所望な電圧印加状態が生じたりしないようにする必要があり、そのために必要な回路構成によって論理的規模の増加を招く。特に、リアルタイム性が必要とされる場合には、必要に応じて、即座に動作可能でなければならない。
【００１３】本発明の目的は、並列処理の並列度が制御可能であって、その並列度に応じた消費電力制御を、論理的規模を増大させず、容易に行なえるデータ処理装置を提供することにある。
【００１４】本発明の前記並びにその他の目的と新規な特徴は本明細書の記述及び添付図面から明らかになるであろう。
【００１５】
【課題を解決するための手段】本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。
【００１６】〔１〕読み込んだ命令を解読して複数の実行部を用いて並列演算処理可能なＣＰＵ等のデータ処理装置において、並列演算処理を行なう場合に、複数個の実行部は命令を解読して制御信号を生成する制御部の指示に応答して動作し、制御部の指示によって個別的に演算動作の停止が指示されるようになっている。動作停止が指示された実行部は、例えば、同期動作用のクロック信号の変化が抑止されて、その演算動作が停止され、且つデータ入出力が禁止される。
【００１７】上記の如く、並列処理を行なう場合に、動作しない実行部のクロック信号を停止すると共に、データ入出力などを禁止することにより、不所望な電力消費を抑制することができる。
【００１８】その場合、実行部での演算処理内容を指定する命令コードそれ自体に、並列実行すべき処理であるか否かを示す情報を持たせる。また、ハードウェアで、並列処理可能な命令を検出している場合には、この検出結果を利用して、並列実行すべき処理であるか否かを検出する。
【００１９】このように、並列実行すべき処理を検出するために、命令コード自体に、その情報を持たせ、また、ハードウェアで、並列処理可能な命令を検出している場合には、この検出結果を利用することにより、随時、消費電力を抑制できる。
【００２０】〔２〕更に、消費電力を制御するために、動作許可する並列処理の数を指定可能にする。これにより、一部の実行部の動作を停止させて、消費電力を低減することができる。
【００２１】具体的には、コンパイラで、使用する並列処理の数を指定可能にし、これに従って、コンパイラは命令コードを生成するようにするとよい。即ち、指定された数より大きい並列処理を実行する命令コードを生成しないようにする。これにより、容易に並列度を指定可能にし、使い勝手を向上することができる。
【００２２】また、ハードウェアで、並列処理可能な命令を検出している場合には、制御レジスタによって使用可能な並列処理の数を指定して、前記ハードウェアはこれを参照して、並列処理の制御を行なうとよい。これにより、プログラムを変更することなく、随時、消費電力の制御を可能にして、使い勝手を向上することができる。プログラムを不所望に変更しないことによって、プログラムの開発効率を向上することができる。
【００２３】〔３〕更には、動作クロック信号を可変にし、少なくとも、並列処理を行なわない程度まで、動作

クロック信号を低下させるようにするとよい。

【００２４】動作クロック信号を可変にする場合には、周辺機能のクロック信号を固定にしつつ、データ処理装置などのバスマスタのクロック信号を変更するようにする。これにより、クロック信号切替え時の処理を最小限にすることができる。

【００２５】

【発明の実施の形態】図１にはデータ処理装置の第１の構成例が示される。同図に示されるデータ処理装置はたとえばＣＰＵ（中央処理装置）１であり、命令リードブロックＦＵ、命令デコードブロックＤＢ、４個の演算処理ブロックＥＸ０～ＥＸ３、汎用レジスタなどのレジスタブロックＲＢ、バッファブロックＢＢから構成される。

【００２６】命令コードＩＳＲは、例えば１２８ビットなどとされ、特に制限されないが、最大４つの処理フィールドｆ０、ｆ１、ｆ２、ｆ３に分割されている。図１の例では、簡単のため、３２ビットづつの均等の処理フィールドｆ０、ｆ１、ｆ２、ｆ３を持つものとする。

【００２７】命令リードブロックＦＵは、前記１２８ビットの命令を、図示されないＲＯＭやキャッシュメモリから、内部データバスＩＤＢを介してリードして、格納し、更に、所定のタイミングで、命令デコードブロックＤＵに転送する。命令のリードアドレスは、プログラムカウンタに基づき、内部アドレスバスＩＡＢに出力される。命令のリード後、プログラムカウンタは更新される。命令リードブロックＦＵは、命令をリードする命令フェッチ部の一例である。

【００２８】命令デコードブロックＤＵは、前記４つの処理フィールドｆ０、ｆ１、ｆ２、ｆ３に対応した部分ｆ０ｄｅｃ～ｆ３ｄｅｃを含んで構成され、前記命令コードをデコードし、対応する演算処理ブロックＥＸ０～ＥＸ３などに対応する制御信号ＣＮＴ０～ＣＮＴ３等を出力する。前記４つの処理フィールドｆ０、ｆ１、ｆ２、ｆ３に対応した部分ｆ０ｄｅｃ～ｆ３ｄｅｃは、無操作を検出する部分ｆ０ｎｏｐ～ｆ３ｎｏｐを含み、この検出結果によって、制御信号としてストップ信号ＳＴＰ０～ＳＴＰ３を出力する。また、レジスタブロックＲＢ、バッファブロックＢＢへも図示を省略する制御信号を出力する。対応する処理フィールドｆ０～ｆ３に、有効な処理の記述が含まれていない場合、或いは無操作を指示する記述が含まれている場合には、命令デコードブロックＤＵはストップ信号ＳＴＰ０～ＳＴＰ３により、対応する演算処理ブロックに動作停止の指示を与える。前記制御信号ＣＮＴ０～ＣＮＴ３は、ストップ信号ＳＴＰ０～ＳＴＰ３以外に、各演算処理ブロックＥＸ０～ＥＸ３に個別に与えられる制御信号を総称している。前記命令デコードブロックＤＵは、前記命令フェッチ部がリードした命令に基づいて前記複数個の実行部の動作を制御する制御部の一例である。

【００２９】演算処理ブロックＥＸ０～ＥＸ３は、算術論理演算器（ＡＬＵ）などを備え、命令デコードブロックＤＵの制御信号に基づいて、動作し、レジスタブロックＲＢ、バッファブロックＢＢと内部バスＤＢ０～ＤＢ３、ＷＢ０～ＷＢ３を介して、データの入出力を行い、所要のデータの演算処理を行なう。それぞれの演算処理ブロックＥＸｉ（ｉ＝０～３）は、命令デコードブロックＤＵの個別の制御信号に基づいて、独立した演算処理を行なう。前記演算処理ブロックＥＸｉは、並列動作可能であって夫々演算器を内蔵した複数の実行部の一例である。

【００３０】対応するストップ信号ＳＴＰｉがイネーブルにされた演算処理ブロックＥＸｉは、クロック信号が停止され、データの入出力が禁止され、内部状態が保持される。演算処理ブロックＥＸｉがパイプライン動作を行なう場合には、ストップ信号ＳＴＰｉは、パイプラインの動作に合わせて、伝播されるように構成する。この詳細な一例は図２に基いて詳述する。

【００３１】レジスタブロックＲＢは、汎用レジスタ、コンディションコードレジスタなどを有し、所用のデータを、内部バスＤＢ０～ＤＢ３、ＷＢ０～ＷＢ３を介して、各演算処理ブロックＥＸｉに出力したり、演算結果を入力したりする。

【００３２】バッファブロックＢＢは、アドレスバッファ、データバッファを備え、内部アドレスバスＩＡＢ、内部データバスＩＤＢを介して、図示はされないＲＡＭやキャッシュメモリとデータの入出力を行い、各演算処理ブロックＥＸ０～ＥＸ３とのインタフェースを採る。

【００３３】なお、命令コードのビット数、演算処理ブロックの数は、データ処理装置の応用範囲、回路規模、処理性能を勘案して最適な数値を選定すればよく、ここで示した数値は一例に過ぎない。例えば、内部バスは複数にしてもよい。内部アドレスバス／内部データバスは、命令コード用とデータ用を分離してもよい。

【００３４】図１のＣＰＵ１によれば、命令コードに含まれる並列演算処理の情報を元にして、命令デコードブロックＤＵがストップ信号ＳＴＰｉによって、必要とされる演算処理ブロックＥＸｉのみを動作させ、必要とされない演算処理ブロックＥＸｉをストップ信号ＳＴＰｉにより自動的に停止することによって、不所望な電力消費を抑止することができる。

【００３５】図２には演算処理ブロックＥＸ０～ＥＸ３に含まれる演算器１０のブロック図の一例が示される。同図に例示される演算器１０は、セレクタ１１、算術論理演算器１２、シフタ・結果バッファ１３、ラッチ回路ＬＴ１～ＬＴ５、及びアンドゲートＡＮＤ１～ＡＮＤ５から構成される。セレクタ１１は、内部バスＤＢ０～ＤＢ３から入力が可能とされ、結果バッファ１３は内部バスＷＢ０～ＷＢ３に出力可能である。

【００３６】演算器１０は、後述の基準クロック信号φ

Ｂから生成されるノンオーバラップの２相クロック信号φ１、φ２で動作する。

【００３７】セレクタ１１は、クロック信号φ１の立ち上がりエッジに同期して、アンドゲートＡＮＤ１を介して制御信号ＣＮＴｉで指示されるデータを入力する。算術論理演算器１２は、クロック信号φ１の立ち上がりエッジに同期して制御信号ＣＮＴｉをラッチ回路ＬＴ１でラッチし、クロック信号φ１の１ステート（１サイクル）で演算を行なう。シフタ・結果バッファ１３は、クロック信号φ１の立ち上がりエッジに同期して制御信号ＣＮＴｉをラッチ回路ＬＴ２でラッチし、その出力をクロック信号φ２の立ち上がりエッジに同期してラッチ回路ＬＴ３でラッチし、ラッチされた制御信号をアンドゲートＡＮＤ５を介して受けることにより、出力動作を行なう。

【００３８】演算器１０に供給されるストップ信号ＳＴＰｉ（ｉ＝０～３）は、クロック信号φ１、φ２の立ち上がりエッジに同期して順次ラッチ回路ＬＴ４，ＬＴ５にラッチされ、その伝播がクロック信号φ１、φ２に同期して順次遅延される。したがって、ストップ信号ＳＴＰｉがイネーブルレベル（この例ではハイレベル）にされると、先ず、クロック信号φ１の立ち上がりエッジに同期したセレクタ１１の入力動作が抑止され、次いで、クロック信号φ１の１ステートに同期される算術論理演算回路１２の演算動作が抑止され、最後に、クロック信号φ２の１ステートに同期される結果バッファ１３の出力動作が抑止される。

【００３９】このように、制御信号ＣＮＴｉの変化を抑止し、データの変化を抑止することによって、消費電力を抑止できる。ＣＭＯＳ半導体集積回路においては、実質的に、当該部分の消費電力を０にできる。演算器は高速で動作するように、消費電力が大きく、データの幅も３２ビットなどと大きくなっており、更に、複雑な演算を行なうために、制御信号の本数も多くなっているから、かかる演算器のデータ及び制御信号の変化を抑止することによって、消費電力を低減することができる。

【００４０】図２の構成より明らかなように、命令、即ち処理プログラムに対応して、１ステート単位、即ち基準クロック信号φＢの単位で、低消費電力制御を行なうことができる。したがって、演算処理ブロックＥＸｉがパイプライン動作を行なう場合に、換言すれば、１命令の演算実行ステージを１クロックサイクルで行なうとき、ストップ信号ＳＴＰｉは、パイプラインの動作に合わせて、伝播されるから、パイプラインを乱すこと無く、若しくはパイプライン上の演算データを不所望に破壊することなく、命令コード単位で演算処理ブロックの並列度を制御して、低消費電力を図ることができる。

【００４１】図３にはデータ処理装置の第２の構成例が示される。同図に示されるデータ処理装置もＣＰＵとされる。図３に示されるＣＰＵ１Ａは、命令リードブロックＦＵ、命令デコードブロックＤＵ、４個の演算処理ブロックＥＸ０～ＥＸ３、リオーダバッファＲＯＲ、リオーダバッファＲＯＭ、汎用レジスタなどのレジスタブロックＲＢ、及びバッファブロックＢＢから構成される。各演算処理ブロックＥＸ０～ＥＸ３は、リザベーションステーションＲＳＶ０～ＲＳＶ３を含む。

【００４２】図３のＣＰＵ１Ａでは、ハードウェアによって並列処理の実行が制御される。即ち、命令デコードブロックＤＵは命令リードブロックＦＵの命令を解析して、並列処理可能な命令を抽出するデコーダアライメントユニットＤＡを持つ。命令デコードブロックＤＵは、デコーダアライメントユニットＤＡで抽出した命令を、並列にデコードして、制御信号ＣＮＴ０～ＣＮＴ３、ストップ信号ＳＴＰ０～ＳＴＰ３を各演算処理ブロックＥＸ０～ＥＸ３に与える。

【００４３】各演算処理ブロックＥＸ０～ＥＸ３のリザベーションステーションＲＳＶ０～ＲＳＶ３は、命令デコードブロックＤＵから出力された制御信号ＣＮＴｉ、ＳＴＰｉを一旦保持し、演算処理ブロックＥＸ０～ＥＸ３の実行状態を監視して、指示された処理の実行が可能になった時点で、演算処理を開始させる。

【００４４】リオーダバッファＲＯＲ，ＲＯＭは、プログラム上の命令の順序と整合性を保つように、並列実行している各演算処理ブロックＥＸｉの演算結果のレジスタブロックＲＢ、バッファブロックＢＢへの書込みタイミングを調整している。

【００４５】図３の例では、並列実行可能な命令が少ない場合には、命令デコードブロックＤＵはストップ信号ＳＴＰｉを、使用しない演算処理ブロックＥＸｉに与える。このストップ信号ＳＴＰｉは、各演算処理ブロックＥＸｉのリザベーションステーションＲＳＶｉで一旦保持され、実行中の処理が終了した時点で有効化される。ストップ信号ＳＴＰｉが有効化されると、演算処理ブロックＥＸｉのクロック信号が停止され、データの入出力が禁止され、内部状態が保持される。演算処理ブロックＥＸｉがパイプライン動作を行なう場合には、ストップ信号ＳＴＰｉが、パイプラインの動作に合わせて、伝播されるように構成されている点は、図１及び図２の場合と同様である。

【００４６】このように、命令の実行状態に基づいて、ハードウェアで解析した並列処理情報を元にして、並列処理可能な数の演算処理ブロックＥＸｉのみを動作させ、その他の演算処理ブロックＥＸｉをストップ信号ＳＴＰｉによって、自動的に停止することによって、不所望な電力消費を抑止することができる。

【００４７】更に、図３の例においては、並列度指定制御信号ＰＡＬＣが、命令デコードブロックＤＵに入力されている。かかる制御信号ＰＡＬＣに基づいて、命令デコードブロックＤＵは並列処理の制御時に、並列動作する演算処理ブロックＥＸｉの数が前記制御信号ＰＡＬＣ

によって指定される数を超えないようにする。前記制御信号ＰＡＬＣは、ＣＰＵ１Ａを採用するマイクロコンピュータ、マイクロプロセッサ、若しくはデータプロセッサ内の適宜のコントロールレジスタの設定値に従って生成される。そのようなコントロールレジスタへのデータ設定は例えばＣＰＵ１Ａ自身がプログラムに従って行なえばよい。

【００４８】図４には図３のＣＰＵ１Ａによる命令実行手順を例示する。図３のＣＰＵ１Ａは有意味の処理を行なう実行命令と、並列処理の制御を行なう制御命令を持つ。制御命令によって、続く命令列の並列実行の制御情報を生成し、命令デコードブロックＤＵはこれを参照して、前記実行命令の並列実行を行なう。制御命令に従って、並列処理する数が少ない場合には、命令デコードブロックＤＵは、前記の通り、所要のストップ信号ＳＴＰｉをイネーブルにする。

【００４９】前記制御命令は、有意味の処理ではないので、ソースプログラム上には記述を必要とせず、コンパイラで自動的に生成すれば都合がよい。

【００５０】図５には本発明に係るデータ処理装置の第３の例であるシングルチップマイクロコンピュータが示される。

【００５１】シングルチップマイクロコンピュータＭＰＵは、全体の制御を司るＣＰＵ２０、データトランスファコントローラ（ＤＴＣ）２１、割込コントローラ（ＩＮＴ）２２、バスコントローラ２３、ＣＰＵ２０の処理プログラムなどを格納するメモリであるＲＯＭ２４、ＣＰＵ２０の作業領域並びにデータの一時記憶用のメモリであるＲＡＭ２５、タイマ２６、ウオッチドックタイマ２７、シリアルコミュニケーションインタフェース（ＳＣＩ）２８、Ａ／Ｄ変換器２９、入出力ポートＩＯＰ１～ＩＯＰ５、入出力ポートＩＯＰＡ～ＩＯＰＦ、クロック発振器（ＣＰＧ）３０、クロック制御回路３１、及びプリスケーラ３２から構成され、公知の半導体製造技術により１つの半導体基板（半導体チップ）上に形成される。ＣＰＵ２０には、特に制限されないが、前述のＣＰＵ１又はＣＰＵ１Ａが採用されている。

【００５２】前記シングルチップマイクロコンピュータＭＰＵは、電源端子として、グランドレベルＶｓｓ、電源電圧レベルＶｃｃ、アナロググランドレベルＡＶｓｓ、アナログ電源電圧レベルＡＶｃｃ、アナログ基準電圧Ｖｒｅｆの入力端子を有し、専用制御端子として、リセットＲＥＳ、スタンバイＳＴＢＹ、モード制御ＭＤ０、ＭＤ１、クロック入力ＥＸＴＡＬ、ＸＴＡＬの各端子を有する。

【００５３】ＣＰＧ３０の端子ＥＸＴＡＬ、ＸＴＡＬに接続される水晶発振子またはＥＸＴＡＬ端子に入力れる外部クロック信号に基づいて、ＣＰＧ３０は基準クロック信号（システムクロック信号）を生成する。基準クロック信号は、バスマスタの基準クロック信号φＢ、周辺回路の基準クロック信号φＳがある。シングルチップマイクロコンピュータＭＰＵの各機能ブロックは基準クロック信号φＳ、φＢに同期して、動作を行う。

【００５４】基準クロック信号φＢは、バスマスタのクロック信号として、ＣＰＵ２０、ＤＴＣ２１、バスコントローラ２３に、更に、外部バスのインタフェース用として入出力ポートＩＯＰＡ～ＩＯＰＦに供給される。クロック信号φＢはクロック制御回路３１によって分周比が選択される。クロック信号φＢの分周比が大きくなれば、これに反比例して、それに同期動作する部分の動作周波数は小さくなり、消費電力も低減される。

【００５５】クロック信号φＳは周辺クロック信号として、タイマ２６、ＳＣＩ２８、Ａ／Ｄ変換器２９、ＩＯＰＡ～ＩＯＰＦ、ＩＯＰ１～ＩＯＰ５に供給される。クロック信号φＳは周波数が固定とされているから、バスマスタの基準クロック信号φＢが変更された場合も、不所望なクロック信号切替え処理などを必要としない。

【００５６】クロック制御回路３１から、ＤＴＣ２１、タイマ２６、ウオッチドックタイマ２７、ＳＣＩ２８、Ａ／Ｄ変換器２９に対して、クロック停止信号すなわちモジュールストップ信号ＭＳＴＰが与えられる。かかるモジュールストップ信号ＭＳＴＰは、クロック制御回路３１から、各機能ブロックに独立した信号が与えられる。クロック停止時には、機能ブロックが内部状態を保持するようにされる。

【００５７】プリスケーラ３２はクロック発振器３０で発生されたクロック信号を分周して所要の回路に供給する。

【００５８】シングルチップマイクロコンピュータＭＰＵの前記機能ブロックは、内部バスによって相互に接続さる。内部バスは、アドレスバス、データバスの他に、リード信号、ライト信号、バスサイズ信号或いはシステムクロック信号などを伝達するコントロールバスを含む。内部データバスは、特に制限されないが、１２８ビット構成とされ、少なくともＲＯＭ２４とＣＰＵ２０との間は、１２８ビットバスでインタフェースされる。特に制限はされないものの、ＲＡＭ２５も同様に１２８ビットバスでインタフェースされる。内部アドレスバスはその位相によって、ＩＡＢ、ＰＡＢの２種類があり、内部データバスもその位相によって、ＩＤＢ、ＰＤＢが存在する。例えば、リードの場合、ＩＡＢの後、ＰＡＢは０．５ステート遅延する。ＰＡＢとＰＤＢは同期している。ＰＤＢの後、ＩＤＢは０．５ステート遅延する。ＩＡＢとＰＡＢ、ＩＤＢとＰＤＢは、バスコントローラ２３によってバッファリングされている。

【００５９】前記機能ブロックやモジュールは内部バスを介して、ＣＰＵ２０によってリード／ライトさる。内蔵ＲＯＭ２４、ＲＯＭ２５は、ＩＡＢ及びＩＤＢでインタフェースされ、１ステートでリード／ライト可能とされる。

11

【００６０】前記シングルチップマイクロコンピュータＭＰＵにリセット信号ＲＥＳが与えられると、ＣＰＵ２０を始めとし、シングルチップマイクロコンピュータＭＰＵはリセット状態になる。このリセットが解除されると、ＣＰＵ２０は所定のアドレスからスタートアドレスをリードして、このスタートアドレスから命令のリードを開始するリセット例外処理を行う。この後、ＣＰＵ２０は逐次、ＲＯＭ２４などから命令をリードし、解読して、その解読内容に基づいて演算処理を行う。

【００６１】割込信号３５は、Ａ／Ｄ変換器２９、タイマ２６、ウオッチドックタイマ２７、ＳＣＩ２８、入出力ポートＩＯＰ５が出力し、割込コントローラ２２はこれを入力して、所定のレジスタなどの指定に基づて、ＣＰＵ２０に割込要求信号３６を与える。これにより、ＣＰＵ２０は実行中の処理を中断して、例外処理状態を経て、割込み要因に応答する所定の処理ルーチンに分岐し、所望の処理を行い、割込要因をクリアしたりする。機器制御においては、かかる割込み（イベント）の発生から、例外処理、所望の処理、割込み要因クリア等の処理を、所定時間内に処理するようにする必要がある。更に、考えられる複数のイベントの競合が発生した場合においても、かかる処理を、所定の時間内に処理する。したがって、通常、割り込み処理ルーチンでは、並列処理を最大にするような命令を配置するようにされる。所定の処理ルーチンの最後には、通常復帰命令が置かれ、この命令を実行することによって前記中断した処理を再開する。

【００６２】図６には前記クロック制御回路３１の一例が示される。クロック制御回路３１は、分周器４０、セレクタ４１、クロック制御レジスタ４２、モジュールストップ制御レジスタ４３、並列度指定制御レジスタ４４、スタイバイ制御回路４５によって構成される。並列度指定制御レジスタ４４はＣＰＵ２０に前記ＣＰＵ１Ａを採用する場合に必要になる。ＣＰＵ２０の前記ＣＰＵ１を採用する場合は不要である。

【００６３】モジュールストップ制御レジスタ４３はモジュールストップを選択可能な機能ブロックに対応して、所定のビット数分だけ設けられている。並列度指定制御レジスタ４４は、少なくとも信号ＰＡＬＣによって通知可能な並列度数分のビット数を有する。

【００６４】クロック制御レジスタは、ＳＳＢＹ、ＭＳＭＥ、ＣＫＳ１、ＣＫＳ０の４ビットを有する。ＳＳＢＹビットはスタンバイ状態への遷移を制御する。ＳＳＢＹビットが１にセットされた状態で、ＣＰＵ２０がＳＬＥＥＰ命令を実行すると、スタンバイ状態に遷移し、ＳＳＢＹビットが０にクリアされた状態で、ＣＰＵ２０がＳＬＥＥＰ命令を実行すると、スリープ状態に遷移する。スタンバイ状態では、シングルチップマイクロコンピュータＭＰＵ全体が停止するが、スリープ状態では、ＣＰＵ２０のみが停止する。

12

【００６５】ＭＳＭＥビットは中速モードの許可ビットであり、このビットが１にセットされている時にのみ、ＣＫＳ１、ＣＫＳ０ビットが有効になる。

【００６６】かかるＭＳＭＥ、ＣＫＳ１、ＣＫＳ０ビットの出力がセレクタ４１を制御し、φＯＳＣ、φＯＳＣ／２、φＯＳＣ／４、φＯＳＣ／８から、バスマスタのシステムクロック信号φＢを選択する。図７にはＭＳＭＥ、ＣＫＳ１、ＣＫＳ０によってシステムクロック信号φＢの選択態様が例示されている。なお、φＯＳＣ以外のクロック信号が選択された状態を中速モードと呼ぶ。

【００６７】クロック発振器３０には、ＥＸＴＡＬ、ＸＴＡＬ端子が入力され、水晶振動子を接続するか、またはＥＸＴＡＬに外部クロック信号を入力する。これらと同一の周波数のクロック信号φＯＳＣを生成する。スタンバイ制御回路４５から、発振器停止信号４６が与えられ、スタンバイ状態ではクロック発振器３０が停止し、クロック信号φＯＳＣはハイレベルで停止するようにされる。

【００６８】分周器４０は、クロック信号φＯＳＣを入力して、順次２分周を行って、クロック信号φＯＳＣ／２、φＯＳＣ／４、φＯＳＣ／８を生成する。

【００６９】セレクタ４１は、クロック信号φＯＳＣ、φＯＳＣ／２、φＯＳＣ／４、φＯＳＣ／８を入力して、クロック制御レジスタ４２のＭＳＭＥビット、ＣＫＳ１、ＣＫＳ０ビットで選択されたクロック信号を、システムクロック信号φＢとして出力し、シングルチップマイクロコンピュータＭＰＵのＣＰＵ２０等のバスマスタに供給する。

【００７０】クロック制御レジスタ４２、モジュールストップ制御レジスタ４３及び並列度指定制御レジスタ４４は、内部データバスＰＤＢに接続され、ＣＰＵ２０からリード／ライト可能にされている。

【００７１】クロック制御レジスタ４２は、ＭＳＭＥビット、ＣＫＳ１、ＣＫＳ０ビットを、セレクタ４１に出力するとともに、ＳＳＢＹビットをスタンバイ制御回路４５に出力する。モジュールストップ制御レジスタ４３は、モジュールストップ信号ＭＳＴＰを、各機能ブロックに供給する。

【００７２】スタンバイ制御回路４５は、例えば、フリップフロップで構成され、ＳＳＢＹビットが１の状態で、ＣＰＵ２０がＳＬＥＥＰ命令を実行すると、かかるフリップフロップがセット状態にされ、発振器停止信号４６を活性状態とし、これによってマイクロコンピュータＭＰＵをスタンバイ状態とする。スタンバイ状態で外部割り込み要求が発生すると、例えば、割り込みコントローラ２２の制御によって、スタンバイ解除信号Ｓ１が活性状態になり、前記フリップフロップはリセット状態になり、発振器停止信号４６が非活性状態になり、スタンバイ状態が解除される。

【００７３】図８には前記シングルチップマイクロコン

ビュータＭＰＵ若しくはＣＰＵ１，ＣＰＵ１Ａの開発環境の概略が示される。シングルチップマイクロコンピュータＭＰＵの使用者は、各種エディタなどを用いて、Ｃ言語乃至アセンブリ言語でプログラムを作成する。これは通常、複数のモジュールに分割して作成される。

【００７４】Ｃコンパイラ５０は、使用者の作成したそれぞれのＣ言語ソースプログラムを入力し、アセンブリ言語ソースプログラム乃至オブジェクトモジュールを出力する。Ｃコンパイラ５０の処理においては、図示はされないプリプロセッサの処理が行われる。

【００７５】アセンブラ５１は、アセンブリ言語ソースプログラムを入力し、オブジェクトモジュールを出力する。

【００７６】リンケージエディタ５２は、上記Ｃコンパイラ５０やアセンブラ５１の生成した、複数のオブジェクトモジュールを入力して、各モジュールの外部参照や相対アドレスなどの解決を行い、１つのプログラムに結合して、ロードモジュールを出力する。

【００７７】ロードモジュールは、シミュレータデバッガ５３に入力され、図示を省略するパーソナルコンピュータなどのシステム開発装置上で、シングルチップマイクロコンピュータＭＰＵの動作をシミュレーションし、実行結果を表示し、プログラムの解析や評価を行なうことができる。また、エミュレータ５４に入力して、実際の応用システム上などで動作する、所謂インサーキットエミュレーションを行ない、マイクロコンピュータ全体としての、実動作の解析や評価を行なうことができる。

【００７８】このほかに、ライブラリアンとして、汎用的なサブルーチンなどを提供することもできる。

【００７９】図９には前記シングルチップマイクロコンピュータＭＰＵのシステム開発装置におけるＣＰＵ１（ＣＰＵ１Ａ）の並列処理選択方法を示す。例えば、図９の（ａ）に示される通り、パーソナルコンピュータなどのシステム開発装置上のディスプレイでプロンプトが表示された状態で、例えば、ＳＥＴ　ＰＡＲＡＬＬＥＬ＝４と入力すればよいようにする。或いは、（ｂ）に示されるように、プロンプトが表示された状態で、ＳＥＴ　ＰＡＲＡＬＬＥＬとコマンドを入力し、これに対して、ＣＰＵの種類及び動作モードのメニューを、例えばＰＡＲＡＬＬＥＬＮｏ．（１．ＰＡＲＡＬＬＥＬ：１、２．ＰＡＲＡＬＬＥＬ：２、３．ＰＡＲＡＬＬＥＬ：３、４．ＰＡＲＡＬＬＥＬ：４）と表示するとともに、メニュー番号の入力を要求し、使用者がメニューの番号１～４のいずれかを入力すればよいようにする。この後コンパイルされるファイルは上記で設定した並列数が使用される。

【００８０】また、（ｃ）に例示されるように、Ｃコンパイラを実行する場合のオプションとして、並列数を設定してもよい。例えば、ｃｃｏｍｐ－ＰＡＲＡＬＬＥＬ＝４ｓｏｕｒｃｅ．ｃと入力する。オプション「－ＰＡＲＡＬＬＥＬ＝４」で４並列を指定し、Ｃ言語ソースプログラムファイル「ｓｏｕｒｃｅ．ｃ」をコンパイルする。

【００８１】このほか、ウィンドウのドロップダウンメニューで選択可能にしてもよいし、ワークステーションなどであれば、Ｃシェルコマンドとして入力することもできる。

【００８２】更に、アセンブラやＣコンパイラなどの、ソースプログラムの制御命令として、ＣＰＵ１の並列処理の数及び動作モードを入力することができる。

【００８３】並列数を少なくした場合には、図１のＣＰＵ１に対応する開発環境では、命令コードの一部が常に、演算処理を示さないようにされる。

【００８４】図３のＣＰＵ１Ａに対応する開発環境では、Ｃコンパイラなどで、並列処理しやすい命令の順序を生成するが、並列数を少なくすれば、その分、Ｃコンパイラの処理が少なくなるので、コンパイル時間の短縮などに寄与できる。また、並列処理制御命令を生成する場合にも同様である。

【００８５】また、並列処理数の異なるＣＰＵ（データ処理装置）を展開した場合にも、Ｃコンパイラなどの開発装置を共通に利用可能にでき、開発環境の開発効率を向上することができる。

【００８６】図１０にはＣＰＵ１，ＣＰＵ１Ａのソースプログラムにおける並列処理選択方法を示す。

【００８７】Ｃ言語のソースプログラムは、コンパイル処理前に、プリプロセッサの処理を受ける。ソースプログラムでは、「＃」で示されるプリプロセッサ文によって、プリプロセッサは処理を行い、コンパイラに指示が行われる。

【００８８】図１０では、Ｃ言語で記述されたソースプログラムであり、プリプロセッサ文（＃ｐｒａｇｍａ）で、コンパイラの処理を指示することができる。（ａ）の場合、

＃ｐｒａｇｍａ　ｐａｒａｌｌｅｌ４（ｋａｎｓｕｌ）
＃ｐｒａｇｍａ　ｐａｒａｌｌｅｌ２（ｋａｎｓｕ２）

で、関数「ｋａｎｓｕｌ」は4並列で、関数「ｋａｎｓｕ２」は２並列でコンパイルすることが指示される。

【００８９】また、（ｂ）の場合、

＃ｐｒａｇｍａ　ｐａｒａｌｌｅｌ２
…
＃ｐｒａｇｍａ　ｐａｒａｌｌｅｌ２ｅｎｄ

で、括られた記述（この場合は、ｗｈｉｌｅループ）が、２並列でコンパイルされる。

【００９０】図１のＣＰＵ１のように、１２８ビットの命令コードに、４つの処理内容が記述される場合、２並列でコンパイルした場合には、所定の２つの演算処理ブロックに対応するフィールドが無操作となる。無操作のフィールドは、命令デコードブロックＤＵによって、対応する演算処理ブロックにストップ信号ＳＴＰｉをアサ

ートさせ、その演算処理動作を停止状態にする。」

【００９１】また、図３のＣＰＵ１Ａのように、並列処理を制御する命令を生成する場合には、２並列でコンパイルすると、３以上の並列処理が指示されないようにされる。

【００９２】図１１には前記シングルチップマイクロコンピュータＭＰＵをプリンタ制御に用いたマイクロコンピュータシステムが例示される。

【００９３】プリンタ制御システムは、シングルチップマイクロコンピュータＭＰＵ、セントロニクスインタフェース或いはユニバーサルシリアルバスなどの受信回路６０、バッファＲＡＭ（ＤＲＡＭ）６１、キャラクタジェネレートＲＯＭ（ＣＧＲＯＭ）６２、プログラムＲＯＭ６３を含み、これらがシングルチップマイクロコンピュータＭＰＵの外部バス６４を介して接続される。

【００９４】また、図１１のプリンタシステムは、更に、印字ヘッド６５、バッファ回路６６、ラインフィードモータ６７、及びキャリッジリターンモータ６８を含み、これらのモータ６７，６８は、それぞれタイマ２６の出力等によって制御される。ラインフィードモータ６７及びキャリッジリターンモータ６８は、特に制限はされないものの、ステッピングモータである。

【００９５】受信回路６０から入力されたデータは、一旦、バッファＲＡＭ６１に格納された後、ＣＰＵ２０によって展開され、バッファＲＡＭ６１から所定の順序で印字ヘッド６５に送られる。この処理量は、印刷するデータによって相違される。例えば、テキストデータはデータ量が比較的少なく、ＣＰＵ２０の処理量も比較的少ない。一方、画像データは、データ量が多く、ＣＰＵ２０の処理量も多い。

【００９６】従って、図１１のシステムにおいて、テキストデータを扱うＣＰＵ２０の処理プログラム（関数）は並列度を低くし、消費電力を低減し、画像データを扱う処理プログラム（関数）は並列度を高くするようにするとよい。並列度を変更することによって、所望の処理能力を維持しつつ、不所望な電力消費を抑止できる。

【００９７】印字ヘッドデータを転送する場合、印字ヘッド６５は印刷精度を向上するため、不規則な形状をしていたり、種々の工夫が試みられ、変更される場合も多いから、ＤＴＣ２１などの固定的なデータ転送では対応が困難で、印字ヘッド６５の仕様に合わせて、ＣＰＵ２０がデータを転送することが必用な場合がある。

【００９８】この場合、ＣＰＵ２０の処理能力によって、印字性能が決る可能性がある。換言すれば、所要の印字性能を実現する為には、上記のデータ転送処理を、所定時間内に終了する必要がある。本発明においては、１つの処理プログラムを、並列の演算器を用いて、高速化しているから、かかる制御に好適である。

【００９９】図５のシングルチップマイクロコンピュータＭＰＵを用いる場合には、ＣＰＵ２０の処理負荷を軽減するために、受信回路６０からバッファＲＡＭ６１へのデータ転送を、ＣＰＵ２０の動作と並行して、ＤＴＣ２１が行ったりすることができる。また、ＤＴＣ２１は、ラインフィードモータ６７及びキャリッジリターンモータ６８を駆動するパルス出力を行う。更に、ＳＣＩ２８の送信データ、受信データの転送を行なう。

【０１００】図１１のプリンタシステムにおいて、図示はされないものの、ＳＣＩ２８はホスト装置などとの通信に使用され、Ａ／Ｄ変換器２９は紙枚数などのセンサ情報を入力する。タイマ２６は、上記モータ駆動の基準タイミングを発生する。これらの基準クロック信号φＳは固定であるから、これらの動作状況に拘らず、ＣＰＵ２０などのバスマスタの基準クロック信号φＢを変更して、随時消費電力を低減することができる。モータの駆動信号は、タイマ２６の基準タイミングに従うから、ＤＴＣ２１によるデータ転送に必要な時間が増えても、基準タイミング周期内であれば問題ない。

【０１０１】周知のように、最近では、プリンタなどのＡＣ電源を使用するものにあっても、消費電力の低減が望まれている。周辺機能の内、使用しないものは、モジュールストップ信号ＭＳＴＰによって、動作を停止させ、消費電力を低減することができる。

【０１０２】半導体集積回路の集積度の向上によって、受信回路６０の一部をシングルチップマイクロコンピュータＭＰＵに搭載して１個の半導体集積回路化することができる。集積度の向上により、配線容量が低減できるから、消費電力の低減にも寄与できる。更に、バッファＲＡＭ６１などの汎用的なメモリも１個の半導体集積回路ＭＰＵに集積することができる。プログラムＲＯＭ６２やＣＧＲＯＭ６３などのように個別のプリンタの機種に応じて変更になるものは、個別の半導体集積回路にする方が都合がよい。

【０１０３】図１２には前記マイクロコンピュータＭＰＵのためのエミュレータが例示される。図１２において７０は図１１のプリンタシステムのような応用システム（ターゲットシステム又はユーザシステムとも称する）、７１はエミュレータ、７２はシステム開発装置である。

【０１０４】エミュレータ７１はエミュレーション用プロセッサ７３を有する。エミュレーション用プロセッサ７３は、応用システム７０に用いられるマイクロコンピュータＭＰＵと同等のマイクロコンピュータ部分にエミュレーション用インタフェースを加えて構成される。エミュレーション用プロセッサ７３の前記マイクロコンピュータ部分はインタフェースケーブル７４を介して、応用システム７０のマイクロコンピュータＭＰＵの実装ソケット７５に結合され、これによってエミュレーション用プロセッサは応用システムを代行制御することができる。

【０１０５】一方、エミュレーション用プロセッサ７３

は前記エミュレーションインタフェースを用いてエミュレーションバス７６に接続される。エミュレーションバス７６のデータバス幅は、ＩＤＢのバス幅に対して１２８ビット等とされる。エミュレーションバス７６には図示はされない状態信号・制御信号の信号線などを含む。前記エミュレーションバス７６を用いて、エミュレーション用プロセッサ７３から、応用システム７０とエミュレーション用プロセッサ７３の内部状態に応じた情報などが出力され、また、エミュレーション用プロセッサ７３に対し、エミュレーションのための各種信号が入力される。

【０１０６】さらに、前記エミュレーションバス７６には、エミュレーションメモリ８１、ブレーク制御回路８２、リアルタイムトレース回路８３などが接続される。前記エミュレーションメモリ８１は、特に制限はされないものの、ＲＡＭなどによって構成され、前記のユーザプログラムを格納した領域と、エミュレーションのためのプログラムを格納した領域とを持つ。前記ブレーク制御回路８２は、エミュレーション用プロセッサ７３による制御状態やエミュレーションバス７６の状態を監視して、その状態が予め設定された状態に達した時に、ブレーク割込み信号ＢＲＫをアサートして、エミュレーション用プロセッサ７３のＣＰＵ２０によるユーザプログラムの実行を停止させ、エミュレーション用プログラム実行状態に遷移させる（ブレークする）。前記リアルタイムトレース回路８３は、前記ＣＰＵ２０のリード動作またはライト動作を示す信号、命令リード動作を示す信号（ＣＰＵステータス信号）、エミュレーションバス７６に与えられるアドレスやデータさらには制御信号を逐次蓄える。

【０１０７】前記エミュレーションメモリ８１、ブレーク制御回路８２、リアルタイムトレース回路８３はコントロールバス８４にも接続され、コントロールバス８４を介してコントロールプロセッサ８５の制御を受けるようになっている。前記コントロールバス８４は、前記コントロールプロセッサ８５に接続されるとともに、ホストインタフェース回路８６を介して、特に制限はされないものの、前記パーソナルコンピュータなどのシステム開発装置７２に接続される。

【０１０８】例えば、システム開発装置７２から入力されたプログラム（ロードモジュール）をエミュレーションメモリ８１のユーザプログラム格納領域に転送し、内蔵ＲＯＭ上に配置されるべきかかるプログラムをＣＰＵ２０がリードすると、エミュレーションメモリ８１上のプログラムがリードされ、実行される。また、ブレーク条件や、リアルタイムトレース条件などもシステム開発装置７２から与えることができる。

【０１０９】前記リアルタイムトレース回路８３は、エミュレーションバス７６のアドレス、データ、制御信号、ＣＰＵステータス信号などを蓄積し、この内容は、所定のコマンドなどでシステム開発装置７２の表示装置に表示して、使用者は、この内容に基いて、プログラムの動作解析を行なう。前記の通り、ＣＰＵ２０のフェッチした命令は、エミュレーションバス７６上に現われ、これを逆アセンブルした、アセンブリ言語でも表示することができる。更に、Ｃ言語ソースプログラムも表示することができる。

【０１１０】前述の通り、ＣＰＵ２０は単一の命令ストリームを実行するようになっているから、ブレークの設定や、トレース情報の取得など、エミュレータ７１による応用システム７０上での動作確認が容易である。また、図１に示されるＣＰＵ１のように、命令コード上に、並列動作の情報が含まれる場合、その情報は、トレース情報として、トレースできるから、動作解析に更に好適である。

【０１１１】上記実施例によれば、以下の作用効果を得るものである。

【０１１２】（１）実行する並列処理の数に対応して、不必要な演算処理ブロックＥＸｉを停止状態にすることにより、不所望な電力消費を抑止することができる。消費電力を低減することにより、不要輻射即ち高周波ノイズを低減することができる。

【０１１３】（２）Ｃコンパイラなどで、命令コードによって、並列度を指定可能にすることで、ＣＰＵの処理状態に応じて、処理性能を優先するか、消費電力抑止を優先するかの選択が可能になる。

【０１１４】（３）並列度指定制御レジスタ４４のような内部Ｉ／Ｏレジスタによって、並列度を指定可能にすることによって、データ処理装置の処理状態に応じて、処理性能を優先するか、消費電力抑止を優先するかの選択が可能になる。

【０１１５】（４）クロック制御レジスタ４２のような内部Ｉ／Ｏレジスタによって、クロック信号を変更することにより、プログラムを変更することなく、ＣＰＵの処理状態に応じて、処理性能を優先するか、消費電力抑止を優先するかの選択が可能になる。これは、命令コードで並列度が指定される場合にも利用できる。一つの関数などが、処理速度が必要な場合と、低消費電力が必要な場合の両方に使用される場合にも、呼び出し側で、クロック信号を変更することによって、処理性能を優先するか、消費電力抑止を優先するかの選択が可能になる。共通の関数とすることにより、プログラムの開発効率を向上できる。周辺クロック信号とは独立に、ＣＰＵなどのバスマスタクロック信号の分周比を可変にすることによって、周辺回路に対する不所望の処理を必要とせず、使い勝手を向上することができる。

【０１１６】（５）Ｃコンパイラなどの開発環境において、並列度を指定可能にすることによって、並列度が異なるデータ処理装置についても開発環境を共通に利用可能にできる。開発環境の開発効率を向上できる。並列度

の拡張を考慮して設計しておく、乃至、予め、上記拡張に対応させておくことにより、更に、開発効率を向上することができる。

【０１１７】（６）Ｃコンパイラにおいてステップ単位、関数単位、ソースプログラムのファイル単位などで、並列度を指定することによって、より使い勝手を向上することができる。

【０１１８】以上本発明者によってなされた発明を実施形態に基づいて具体的に説明したが、本発明はそれに限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能であることは言うまでもない。

【０１１９】例えば、並列処理の数は任意にできる。演算処理ブロックは全て同じ機能である必要はない。１ステート演算用、ロード／ストア処理用など個別の処理機能を持たせてもよい。並列処理の方法も、ＶＬＩＷやスーパスカラなどのほか、ＭＩＭＤ（マルチプル・インストラクション・マルチプル・データ）などであってもよい。

【０１２０】データ処理装置の詳細、並列処理の制御の方法や、演算処理ブロックの詳細も、任意に変更可能である。

【０１２１】また、命令コードの基本単位１２８ビットに限定する必要はなく、いわゆるＶＬＩＷ型のデータ処理装置にあっては、２５６ビットや６４ビットでもよく、いわゆるスーパスカラ型のデータ処理装置にあっては、１６ビット或いは３２ビットなど任意のビット幅とできる。データバスの幅も任意にできる。

【０１２２】シングルチップマイクロコンピュータのその他の機能ブロックについても何等制約されない。データ処理装置の実現方法としては、シングルチップマイクロコンピュータや半導体集積回路には限定されない。

【０１２３】マイクロコンピュータシステムとしてもプリンタに限定されない。携帯型機器、例えばオートフォーカスのカメラなどであってもよい。焦点を合わせるまでは並列度を高くし、一旦焦点が合った後の被写体の動きに焦点を追従させるときには並列度を低くすることが考えられる。

【０１２４】開発環境の詳細についても、任意にできる。少なくとも、ユーザがプログラムを作成する際に、並列度を指定できるようにすればよい。

【０１２５】以上の説明では主として本発明者によってなされた発明をその背景となった利用分野であるシングルチップマイクロコンピュータに適用した場合について説明したが、それに限定されるものではなく、その他のマイクロコンピュータやデータプロセッサなどその名称の如何に拘わらずデータ処理装置に広く適用可能である。

【０１２６】

【発明の効果】本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。

【０１２７】すなわち、並列処理を行なう場合に、演算処理ブロックの数より、並列実行可能な処理が少ない場合に、動作を要しない演算処理ブロックのクロック信号を停止させると共に、データ入出力などを禁止することにより、不所望な電力消費を抑止することができる。

【０１２８】動作許可する並列処理の数を、コンパイラや制御レジスタで、指定可能にすることによって、一部の並列処理ブロックの動作を停止させて、消費電力を低減することができる。

【０１２９】制御レジスタでクロック信号を変更することによって、プログラムを変更することなく、データ処理装置の処理状態に応じて、処理性能を優先するか、消費電力抑止を優先するかの選択が可能にできる。

【図面の簡単な説明】

【図１】本発明に係るデータ処理装置の第１の例としてＣＰＵを例示するブロック図である。

【図２】図１のＣＰＵに含まれる演算処理ブロックの演算器を例示するブロック図である。

【図３】本発明に係るデータ処理装置の第２の例として別のＣＰＵを示すブロック図である。

【図４】図３のＣＰＵによる命令実行手順を例示する説明図である。

【図５】本発明に係るデータ処理装置の第３の例としてシングルチップマイクロコンピュータを例示するブロック図である。

【図６】シングルチップマイクロコンピュータに内蔵されるクロック制御回路を例示するブロック図である。

【図７】ＭＳＭＥ、ＣＫＳ１、ＣＫＳ０によるシステムクロック信号φＢの選択態様を例示する説明図である。

【図８】ＣＰＵの開発環境を概略的に示す説明図である。

【図９】ＣＰＵのシステム開発装置における並列処理選択方法を示す説明図である。

【図１０】ＣＰＵのソースプログラムにおける並列処理選択方法を示す説明図である。

【図１１】図５のシングルチップマイクロコンピュータをプリンタ制御に用いたマイクロコンピュータシステムのブロック図である。

【図１２】図５のマイクロコンピュータのためのエミュレータを例示するブロック図である。

【符号の説明】

１，１Ａ，２０　ＣＰＵ
ＦＵ　命令リードブロック
ＤＵ　命令デコードブロック
ＤＡ　デコーダアライメントユニット
ＥＸ０～ＥＸ３　演算処理ブロック
ＲＳＶ０～ＲＳＶ３　リザベーションステーション
ＲＢ　レジスタブロック
ＢＢ　バッファブロック

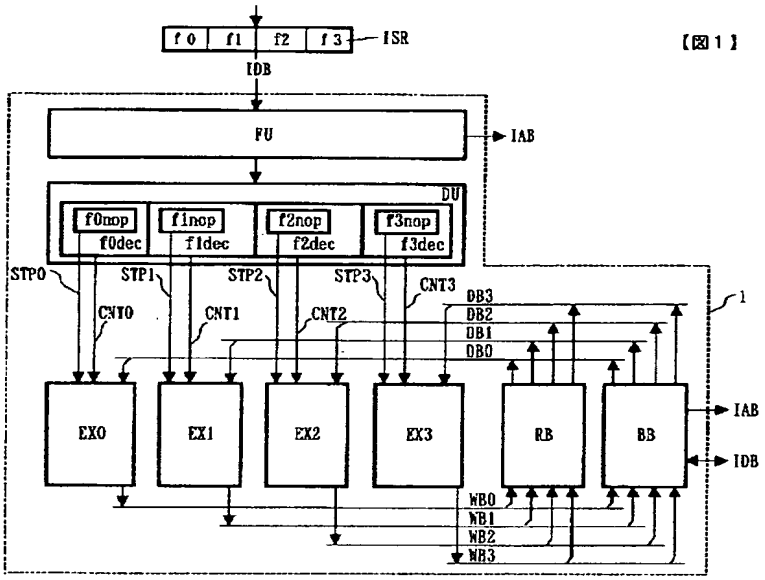
21

ＳＴＰ０～ＳＴＰ３（ＳＴＰｉ）　ストップ信号
ＣＮＴ０～ＣＮＴ３（ＣＮＴｉ）　制御信号
φ１、φ２　ノンオーバラップ２相クロック信号
φＢ、φＳ　基準クロック信号
ＰＡＬＣ　並列度指定制御信号
１０　演算器

22

＊１１　セレクタ
１２　算術論理演算回路
１３　シフト回路・結果バッファ
３０　クロック発振回路
３１　クロック制御回路
＊　４４　並列度指定制御レジスタ

【図１】



【図１】

【図７】

【図７】

MSME	CKS1	CKS0	φB
0	-	-	φOSC
1	0	0	φOSC
	0	1	φOSC/2
	1	0	φOSC/4
	1	1	φOSC/8

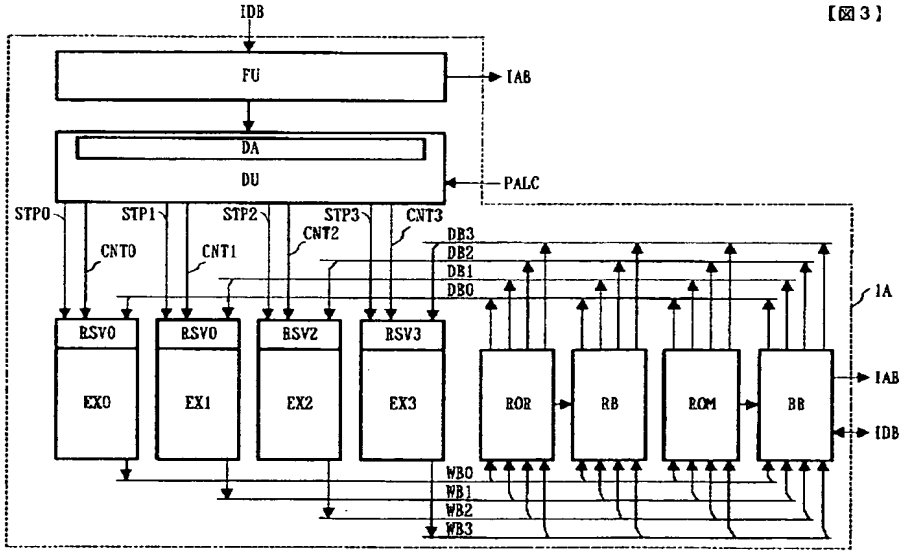
【図１０】

【図１０】

(a)

```
void kausu1(void);
void kausu2(void);
#pragma parallel4(kansu1)
#pragma parallel2(kansu2)
...
void kausu1(void)
        {
        ...
        }
...
void kausu2(void)
        {
        ...
        }
...
```

【図３】

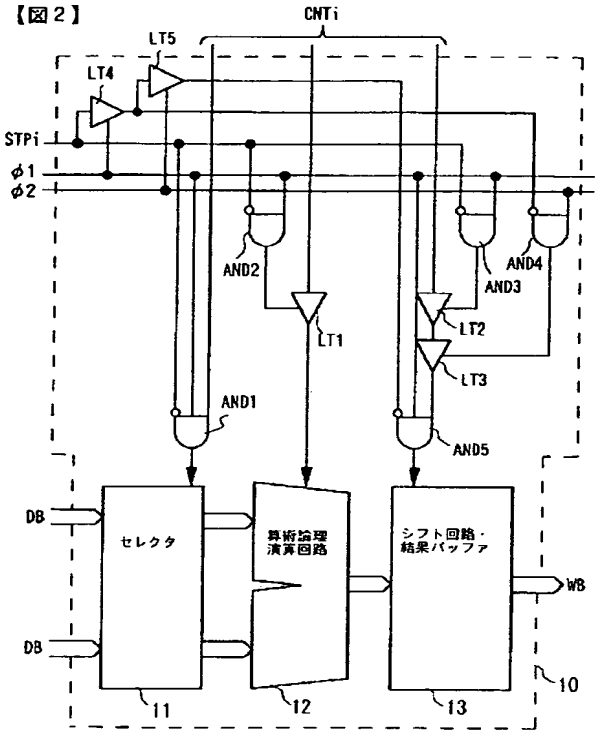


【図３】

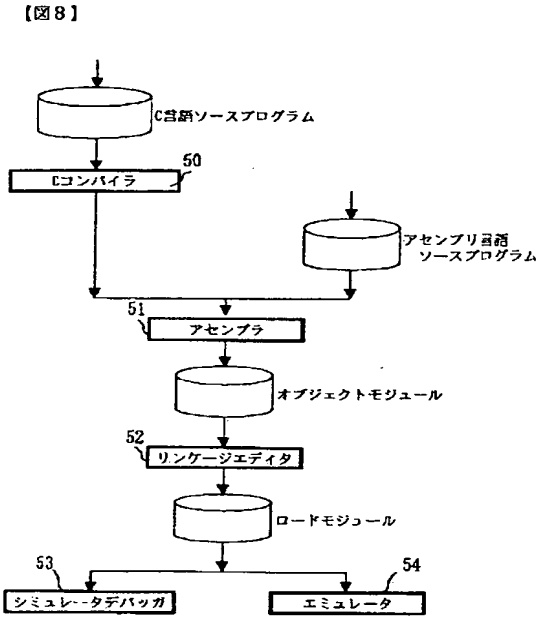
(b)

```
void kausu1(void);
...
void kausu1(void)
        {
        ...
        #pragma parallel2
        while (...){
        ...
        }
        #pragma parallel2end
        ...
        }
...
```

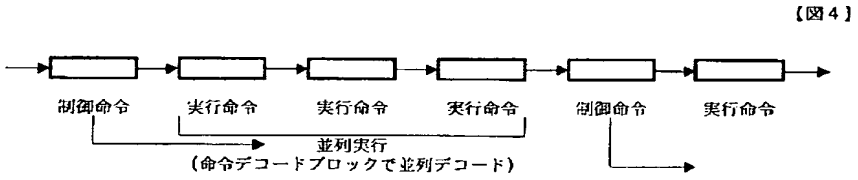
【図２】


【図２】
CNTi
LT5
LT4
STPi
φ1
φ2
AND2
AND4
AND3
LT1
LT2
LT3
AND1
AND5
DB
セレクタ
算術論理演算回路
シフト回路・結果バッファ
WB
DB
10
11
12
13

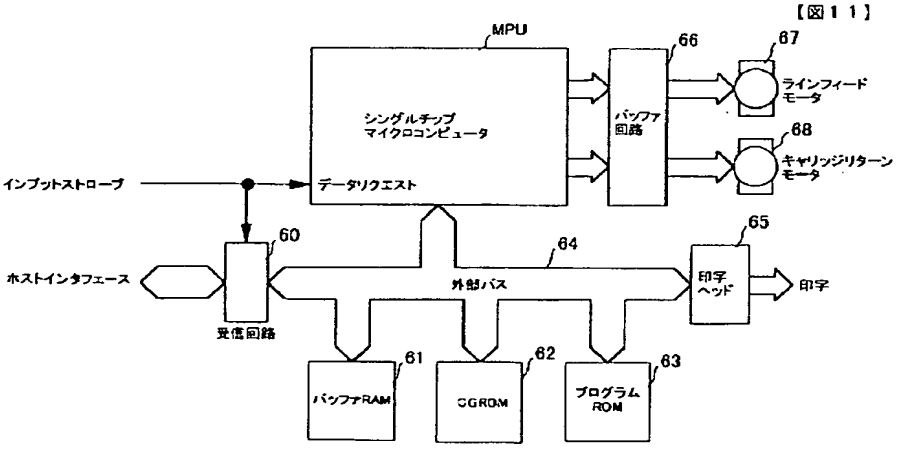
【図８】


【図８】
C言語ソースプログラム
Cコンパイラ
50
アセンブリ言語ソースプログラム
51
アセンブラ
オブジェクトモジュール
52
リンケージエディタ
ロードモジュール
53
54
シミュレータデバッガ
エミュレータ

【図４】


【図４】
制御命令
実行命令
実行命令
実行命令
制御命令
実行命令
並列実行
(命令デコードブロックで並列デコード)

【図１１】


【図１１】
MPU
シングルチップマイクロコンピュータ
データリクエスト
66
バッファ回路
67
ラインフィードモータ
68
キャリッジリターンモータ
インプットストローブ
60
ホストインタフェース
受信回路
64
外部バス
65
印字ヘッド
印字
61
バッファRAM
62
CGROM
63
プログラムROM

【図５】

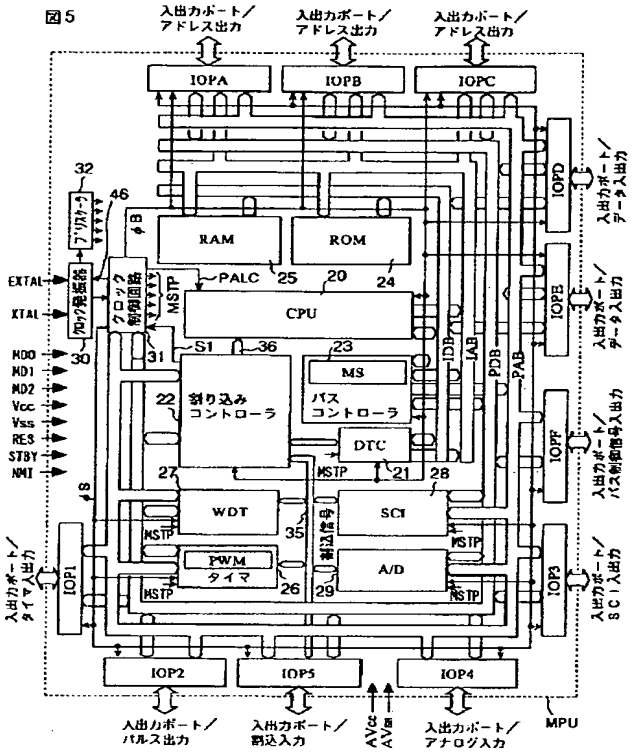
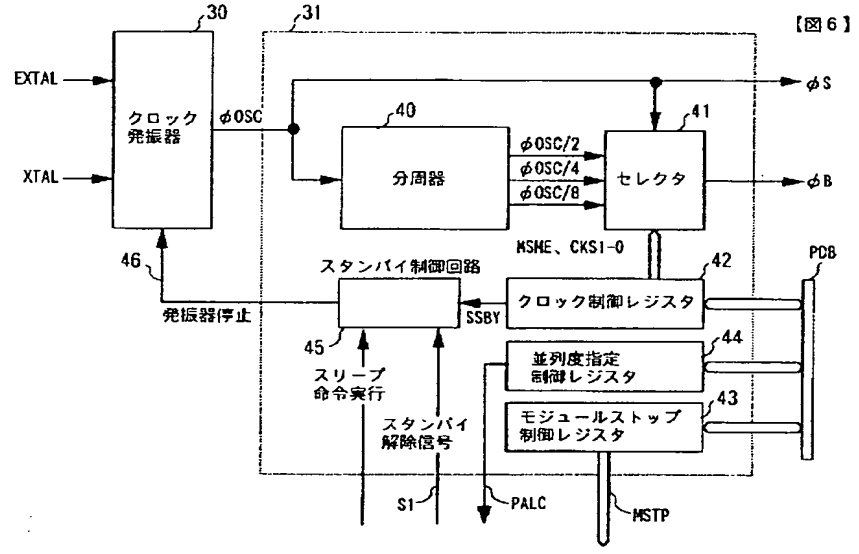
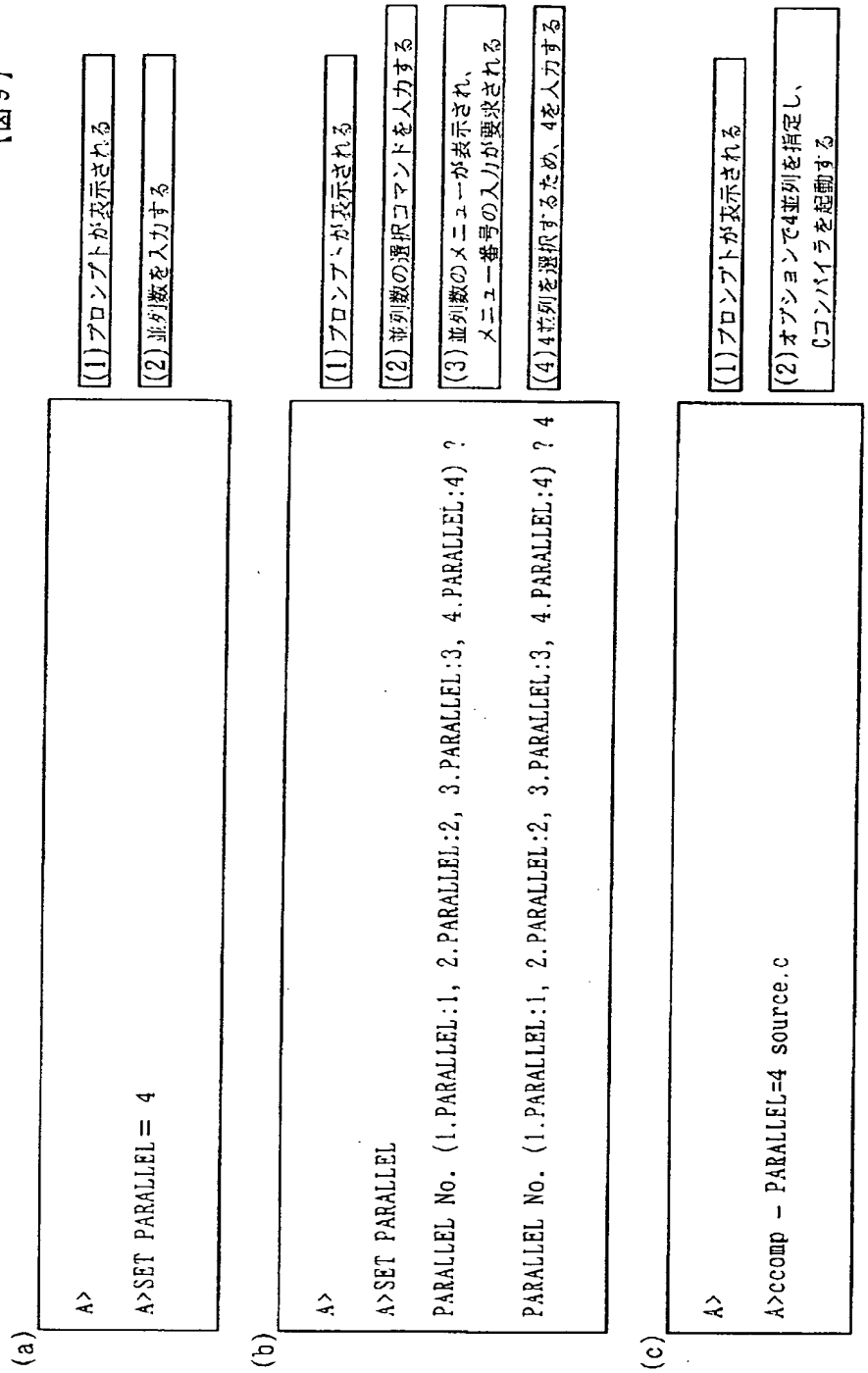


図5
入出力ポート／アドレス出力
IOPA
IOPB
IOPC
IOPD
入出力ポート／データ入出力
RAM
ROM
CPU
PALC
MSTP
クロック発振器
クロック制御回路
EXTAL
XTAL
MD0
MD1
MD2
Vcc
Vss
RES
STBY
NMI
割り込みコントローラ
バスコントローラ
MS
DTC
WDT
SCI
PWM タイマ
A/D
IOP1
IOP2
IOP5
IOP4
IOP3
入出力ポート／タイマ入出力
入出力ポート／パルス出力
入出力ポート／割込入力
入出力ポート／アナログ入力
入出力ポート／SCI入出力
MPU

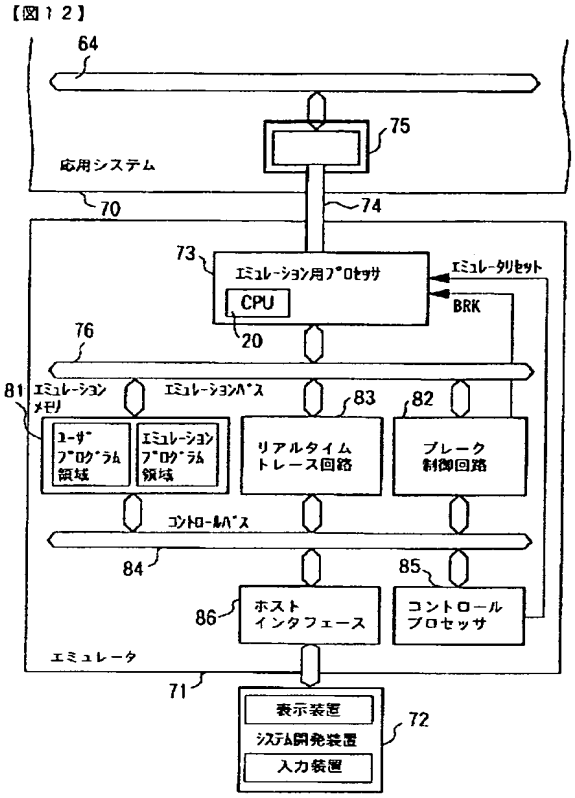
【図６】


【図6】
30
31
EXTAL
XTAL
クロック発振器
φOSC
φS
φB
40
分周器
φOSC/2
φOSC/4
φOSC/8
41
セレクタ
MSME、CKS1-0
42
PDB
46
スタンバイ制御回路
発振器停止
SSBY
クロック制御レジスタ
45
スリープ命令実行
並列度指定制御レジスタ
44
モジュールストップ制御レジスタ
43
スタンバイ解除信号
S1
PALC
MSTP

【図9】


【図9】
(a)
A>
A>SET PARALLEL = 4
(1)プロンプトが表示される
(2)並列数を入力する
(b)
A>
A>SET PARALLEL
PARALLEL No. (1.PARALLEL:1, 2.PARALLEL:2, 3.PARALLEL:3, 4.PARALLEL:4) ?
PARALLEL No. (1.PARALLEL:1, 2.PARALLEL:2, 3.PARALLEL:3, 4.PARALLEL:4) ? 4
(1)プロンプトが表示される
(2)並列数の選択コマンドを入力する
(3)並列数のメニューが表示され、メニュー番号の入力が要求される
(4)4並列を選択するため、4を入力する
(c)
A>
A>ccomp - PARALLEL=4 source.c
(1)プロンプトが表示される
(2)オプションで4並列を指定し、Cコンパイラを起動する

【図１２】



フロントページの続き

(51)Int.Cl.'	識別記号	F I		テーマコード (参考)
Ｇ０６Ｆ 9/45		Ｇ０６Ｆ 9/44	３２２Ｆ	

Ｆターム(参考) 5B011 DA06 EA08 EA09 LL08 LL11
LL12
5B013 DD01 DD04
5B033 AA05 AA13 AA14 BC01 BD01
BE05 DD05
5B079 AA07 BA03 BA12 BA13 BB01
BC01 DD03
5B081 AA06 AA07 CC32